



## How to / Maniaplanet / Manialink / Build a Manialink

Author : Marcel @ Funtrackers



LAST PAGE

PAGE

1

2

3

4

5

6

7

8

9

NEXT PAGE



### 1.1 XML crash course

ManiaLinks make use of the XML standard to specify the elements to be displayed when viewing them. If you are already familiar with HTML, you know the basic syntax used for XML and so for ManiaLinks, but there are small differences between HTML and the stricter XML.

So what exactly is XML? The three letters stand for eXtensible Markup Language, and is a general syntax for describing various documents. Although XML is standardized it sets no restrictions to the actual elements to be used: The application itself decides which elements are needed, and what they should represent in the end. In this part of the Tutorial I will only describe these features of XML which you really have to know to create a valid ManiaLink file.

#### Hello World: Creating your first ManiaLink file

An XML file is simply a text file, which commonly ends with .xml to indicate it to be XML. This means, you may use whatever text editor you want, nevertheless, especially for newcomers an editor with Syntax Highlighting is recommended, as this feature directly reveals most errors as of wrong colorization of the following code. I myself recommend the editor Notepad++, a lightweight editor supporting syntax highlighting for many different languages. In the following I assume you use Notepad++, but all steps may be adapted to other editors as well.

After installing and opening Notepad++, select in the Language menu XML, to enable the correct highlighting, then type the following into it:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<manialink version="1">
<label text="Hello World!" />
</manialink>
```

Before we finally save this file, we have to ensure it has the right encoding. This is important, if you use special characters like German umlauts, because if you save the file in the wrong encoding, these characters will be messed up and not displayed correctly on the ManiaLink. In Notepad++ open the Encoding menu, and check what encoding the file currently has (i.e. which element is marked). If it is not, encode in **UTF-8 without BOM**, click the option Convert to UTF-8 without BOM a little more down in the menu. Double check the correct encoding to be on safe side.

When saving the file, you have to pay attention to really save it as XML file (i.e. with ending .xml), because Notepad++ then will automatically detect it as XML and select the correct highlighting for you. You may either input the file extension directly in the file name field, or you may select eXtensible Markup Language in the file type dropdown field. For now, simply name your file helloworld.xml and save it into ManiaPlanet\Media\Manialinks\ within your Documents directory.

Now go into ManiaPlanet, and enter file://Media/Manialinks/helloworld.xml into the address bar which shows up when you move your mouse to the top of the screen – If you now see a white Hello World! printed on a gray background, you successfully created your first ManiaLink Biggrin (If you want to close the ManiaPlanet browser, click on the left most button in the top bar or press the Esc key.)

**Hint:** The file:// method of accessing ManiaLinks only works locally on your machine; nobody else will be able to see your ManiaLink this far. If you upload this file on a webspace, you can call it with its URL as known from normal websites. Before learning how to register a code for a ManiaLink file, we first need to learn how XML in detail works

#### A closer look onto the syntax: The XML header

If you look on the example from above, you see as first line the so called XML header. This header always introduces an XML file, and tells the application that it actually is XML. The values specified in the header are like a little configuration of the XML file. This line will mostly never change at all, so either remember the concrete values, or copy&paste this line in each new XML file you create.

The reason simply is: There is currently no other version than 1.0, using UTF-8 as encoding is the best choice to make your ManiaLink compatible to all languages, and what in detail standalone tags are will be described later.

**Hint:** It is very important that the very first character of a file is the < of the XML Header. You are neither allowed to add any spaces or line breaks, nor any other characters in front of it.

#### The opening and closing of Tags

So what about the other three lines of the example? These are the actual XML file, describing our ManiaLink we want to show in ManiaPlanet. Now we see the so called Tags: They always start with an opening angle bracket < and end with a closing one >, everything between these brackets belongs to the tag. At least, the name of the tag is specified. In our example we have two manialink tags and one label tag.

As you may see, there are in total three different kinds of tags: An opening tag has no slash / in it, and never comes alone. Whenever you have an opening tag like the first manialink one, you will always find its corresponding closing tag: This tag has the same name as the opening one, but has a leading / before the actual name, as seen in the second manialink tag.

In addition, an opening tag may have so called attributes, specified either as name="value" or name=value'. If you use single quotes or double quotes does not really matter; there is no difference between them at all. Back to our example, we can see, that the opening manialink tag has an attribute named version, which value is set to 1. The label tag, too, has an attribute, here it is called text and its value is Hello World!, which we already saw printed in ManiaPlanet. Opening tags may have several attributes, but they may also have none. Closing tags never have attributes at all.

The third type of a tag is more a shortcut than really a new type. Look at the label tag in the above example: This one has the slash / at the end of the tag, and therefore is called a standalone tag. It is simply a short form of <label text="Hello World!"></label>, and can always be used if there is nothing else between the opening and closing tag.

An opening tag with its corresponding closing tag together with everything between them is called a node of the XML document. A node always starts with the opening tag, and ends with a closing tag with the same name, and everything between these tags is called the content of the node. (In case of a standalone tag, this tag alone forms the node, of course.) The content now can be either of simple text, or, what we will mostly see in ManiaLinks, other nodes, i.e. other opening and closing tags. Important is, that a node always is completely contained in another node, where the outer node is called the parent and the inner one the child node. Back to the example, we have the label node as a child of the manialink node. The label node itself has no content, that's why we were able to use the shorter syntax as standalone tag.

#### Special tags: Root node, comments and again the XML header

The manialink node now is a special node: It has no parent, and therefore is called the root node. A valid XML document must always have exactly one root node, and as this node then holds all data of the document as child nodes, it is also called the document node. In ManiaLinks, the root node always is a <manialink>.

Another special kind of nodes are comment nodes. You can place a comment wherever you can place other nodes, so keep attention to placing it neither outside the root node, nor inside a certain tag e.g. between the attributes. As the name says, you are able to add additional comment to your XML file, e.g. to say which role this certain part of file has in the final ManiaLink, and the XML parser will per definition ignore any comments within the file. The syntax of a comment is slightly different than of the other tags:

```
<!-- This is a comment -->
```

As you can see, a comment starts with < for a tag followed by an exclamation mark and exactly two hyphens, and it ends with again exactly two hyphens and the closing >.

**Hint:** You may use any characters within comments you want. There is only one exception: You are not allowed to use -- inside a comment, as this would end the comment and requires the closing > to directly follow. So whatever you print as comment, never print -- in it.

The last special tag is the XML Header, which always is the very first tag in an XML file and the only exception which is written outside the root node.

#### Special characters and their escaping

Wherever you have characters defining a syntax, you always have the problem in printing these characters themselves. This is the same case in XML: The angle brackets or quotes may disturb the XML syntax, making your complete file invalid. Of course there is a way of making these characters safe, the so called escaping of the characters. In total, only five characters are affected by this problem:

< becomes &lt; as in "Lower Than" – Escape outside of tags.  
> becomes &gt; as in "Greater Than" – Escape outside of tags.  
" becomes &quot; as in "QUOTE" – Escape in double quoted attribute values.  
' becomes &apos; as in "APOSrophe" – Escape in single quoted attribute values.  
& becomes &amp; as in "AMPersand" – Escape always.

As you can see, the escaping sequence always starts with an ampersand & (which makes this character a very harmful one Biggrin), followed by an entity and finalized with a semicolon ;. You may always escape these five characters when they are not part of the syntax, but you only must escape them if they occur at the places as specified above. Comments are an exception of this escaping: You don't have to escape anything within a comment, as it gets ignored by default.

**Hint:** If you use a PHP generated ManiaLink and want to output the value of a user input, make sure to escape these characters! Otherwise, a simple " or ' may make your ManiaLink invalid and thus not available to anyone anymore. **Warning:** If you use single quotes for attribute values, make sure to use htmlspecialchars() properly to escape the single quotes, i.e. you have to use the following line:

```
$escapedText = htmlspecialchars($textToBeEscaped, ENT_QUOTES, 'UTF-8');
```

#### Little Test: Find the errors

What do you think: Are you already able to write a valid XML file? Let's have a little test: The following code shows an invalid XML file. Can you find all of the errors, which were made in this file?

##### Invalid XML file:

```
<root>
<!-- This is a negative example of XML -- Do not copy&paste it! -->
<child1><child2>Some content</child1></child2>
<input type="checkbox" checked>
<label link="http://www.example.org/?param1=value1&param2=value2" />
</root>
<standalone />
```

##### Errors made ... here's the solution

###### Missing XML Header

Although a missing XML header does not hinder ManiaPlanet or other XML parsers to correctly parse the file, it is not the clean way of defining an XML file. You can see this the same way as if you would name an JPG with .img: Some programs won't have any problems in opening the JPG, but e.g. Windows itself fails to display a thumbnail for it.

###### Root node is not alone

The first node of a document (except the XML header) is always the root node, in our erroneous example it is <root>. A valid XML file is allowed to have exactly one root node, so the <standalone> tag is not allowed to be placed outside the <root>.

###### Wrong interlacing of <child1> and <child2>

The nodes <child1> and <child2> are not nested correctly: Neither of these tags completely contains the other one. You have either to close <child2> before <child1>, or open <child2> before <child1> to have one containing the other. Or close <child1> before opening <child2> to make them both childs of the root tag.

###### Missing closing tag of <input> and attribute without value

The <input> tag is neither a standalone tag, nor can we find any corresponding closing tag. Furthermore, the attribute checked does not have any value. This tag should show the differences between HTML and the stricter XML: Whereas this tag is absolutely well formed in HTML, XML itself would never accept it.

###### Character & not escaped to &amp;

The character & must always be replaced with &amp; to keep the XML file valid. No exceptions. Except in comments Biggrin

###### Unallowed "-" in Comment

You are not allowed to write two hyphens -- in the middle of a comment, as those indicate the end of it. Remember: The & character need not to be escaped in a comment, and thus is not an error in this case.

**Hint:** If you want to check the correctness of an XML file, you can simply open it in a normal browser like Firefox or Chrome. (Do not use the Internet Explorer, it cannot help you at all ^^) As long as the browser recognizes that the file is an XML file, it will try to parse the XML and print the document tree. If there is any syntax error in the file, the browser of course will fail to parse it, and will print the first error where the file got invalid.



LAST PAGE

PAGE

1

2

3

4

5

6

7

8

9

NEXT PAGE

